VI International Conference on Particle-based Methods – Fundamentals and Applications
PARTICLES 2019
E. Oñate, M. Bischoff, D.R.J. Owen, P. Wriggers & T. Zohdi (Eds)

PARALLEL IMPLEMENTATION OF FAST METHODS FOR VORTEX INFLUENCE COMPUTATION IN VORTEX METHODS FOR 2D INCOMPRESSIBLE FLOWS SIMULATION

DARIA D. LEONOVA[1], ILIA K. MARCHEVSKY[1,2] AND EVGENIYA P. RYATINA[1,2]

[1] Bauman Moscow State Technical University
2-nd Baumanskaya st., 5, 105005 Moscow, Russia

[2] Ivannikov Institute for System Programming of the Russian Academy of Sciences
Alexander Solzhenitsyn st., 25, 109004 Moscow, Russia

daria.denisovna9713@gmail.com, iliamarchevsky@mail.ru, evgeniya.ryatina@yandex.ru

Key words: Vortex Methods, Incompressible Media, Vortex Influence Calculation, Fast Methods, Barnes — Hut-type Method, Fast Fourier Transform, OpenMP, MPI

Abstract. Vortex methods are a powerful tool for solving engineering problems of incompressible flow simulation at small subsonic speeds. The main idea is to consider vorticity as a primary computed variable. Vorticity distribution is simulated by a set of elementary vorticity carriers — vortex elements. Their velocity in the flow is a sum of the convective and diffusive ones. The simplest way to compute the convective velocity of each vortex element is to summarize the influences of all the other vortices, it should be done at every time step. Such problem is similar to the N-body gravitational problem, its computational complexity is proportional to N^2 (N is number of vortices). This fact restricts significantly the applicability of vortex methods.

Two approximate fast methods of logarithmic ($N \log N$) computational complexity are implemented and investigated. The first method is analogous of the Barnes — Hut fast method for the gravitational N-body problem; the second one is based on the possibility of convolution integral fast calculation through Fast Fourier Transform (FFT) technique with further results correction, which permits to take into account the influence of closely-spaced vortices. Sequential and parallel implementations of both methods are developed. Numerical experiments show that the FFT-based method is more efficient in comparison to the Barnes — Hut method; it provides the acceleration of about 1000 times for the velocities calculation for $N = 500\,000$ vortex elements (in comparison to the direct "point-to-point" calculation). The number of mesh cells doesn't effect the method accuracy, however it determines the computational complexity of the algorithm. It is found that the mesh size should be chosen according to the derived estimation of the algorithm's numerical complexity and available computational resources.

1 INTRODUCTION

For many problems of two-dimensional outer gas and fluid flows simulation the Lagrangian vortex methods [1, 2] can be very efficient in comparison to well-known mesh methods. Their range of applicability is limited by incompressible flows, however for some engineering applications the compressibility can be neglected. We consider pure Lagrangian meshless modification of vortex methods, namely Viscous Vortex Domains method [3]; its main idea is considering the vorticity as a primary computed variable. The vorticity in the flow moves with the velocity which is a sum of the convective velocity and diffusive one caused by the viscosity influence. The vorticity distribution $\boldsymbol{\Omega} = \Omega \boldsymbol{k}$ is simulated by a set of elementary vorticity carriers — vortex elements, characterised by their positions in the flow domain $\boldsymbol{r}_i$ and circulations Γ_i, which remain constant:

$$\Omega(\boldsymbol{r}) = \sum_{i=1}^{N} \Gamma_i \delta(\boldsymbol{r} - \boldsymbol{r}_i),$$

where N is number of vortex elements, which simulate the vortex wake, $\delta(\boldsymbol{r})$ is two-dimensional Dirac delta-function, $\boldsymbol{k}$ is the unit vector orthogonal to the flow plane.

The convective velocity of the vortex elements is calculated through known vorticity field and incident flow velocity $\boldsymbol{V}_\infty$ according to the Biot — Savart law (we consider flows without streamlines surfaces, however all the presented results can be transferred to more general cases):

$$\boldsymbol{V}_{conv}(\boldsymbol{r}, t) = \boldsymbol{V}_\infty + \int\limits_S \underbrace{\frac{\boldsymbol{k} \times (\boldsymbol{r} - \boldsymbol{\xi})}{2\pi |\boldsymbol{r} - \boldsymbol{\xi}|^2}}_{\boldsymbol{Q}(\boldsymbol{r} - \boldsymbol{\xi})} \Omega(\boldsymbol{\xi}, t) dS_\xi = \boldsymbol{V}_\infty + \sum_{i=1}^{N} \Gamma_i \boldsymbol{Q}(\boldsymbol{r} - \boldsymbol{r}_i). \qquad (1)$$

The vortex influence calculation by direct summation according to (1) is the most time-consuming operation in the vortex method algorithm [4]. The computational complexity is proportional to N^2, and is similar to the gravitational N-body problem. In practice the number of elements N can reach 10^5, so it takes about 10^{10} operations only for the sum (1) calculation. Note, that such sum should be calculated at every time step while the number of steps can has order of tens thousands. So, the direct calculation of the sum (1) becomes impossible in a reasonable time.

This problem can be partially solved using the modern graphic accelerators (GPU). As for all the particle methods, implementation of the vortex methods by using the Nvidia CUDA technology is very efficient [4]. However this approach doesn't solve the mentioned problem fundamentally, because the computational complexity remains squared and computations for more than $3 \cdot 10^5$ vortex elements again require unacceptable time.

The computational complexity of the problem can be reduced significantly by implementing of the approximate fast methods. In this paper we consider the Barnes — Hut-type method [5], initially developed for N-body problem, and the method based on the fast Fourier transform and further correction procedure [7]. Both methods have logarithmic computational complexity (proportional to $N \log N$); their sequential and parallel implementations are developed.

2 THE BARNES — HUT-TYPE METHOD

The main idea of this method is that the influence of the groups of closely adjacent vortex elements on another such groups located far apart, can be calculated approximately using linearized formulae [5]. For this purpose the hierarchical tree-structure of rectangular space domains (cells) is constructed in the flow domain. The zero-level cell contains all the vortex elements, and then it is divided across its long side into two first-level cells, each is reduced horizontally and vertically according to its vortices in order to exclude empty area. Next, similarly the second-level cells are constructed, etc. Such procedure is continued until the target level is achieved or the cell contains single vortex. The whole algorithm consists of the following stages:

1. Zero-level cell formation which contains all the vortex elements.
2. Tree structure construction.
3. Calculation of the necessary tree-cells parameters (centers of positive and negative vorticity and total circulations).
4. For every terminal tree-cell the following operations are performed:
   a) tree traversal and determination of the far-spaced cells according to chosen proximity criteria;
   b) accumulation of the linear expansion coefficients for all far-spaced cells;
   c) exact calculation of the influence from the vortices in cells from neighboring zone according to (1);
   d) summation of the influences calculated approximately and exactly.

The numerical experiments were performed for different time-consuming problems (Fig. 1) and the results (time of computations) are in a good agreement with theoretical estimation (logarithmic computational complexity).

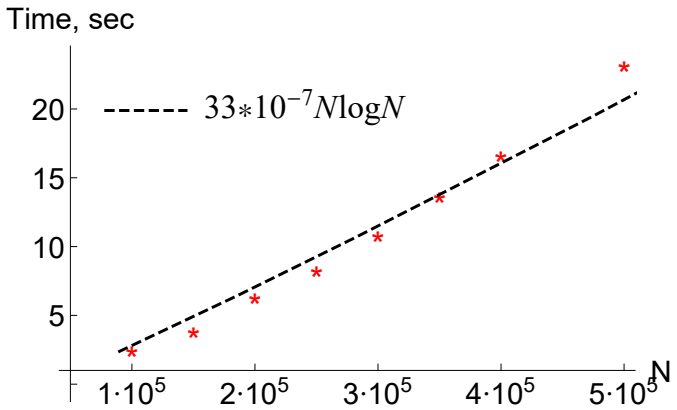


Figure 1: Time of computations for different number of vortex elements N

The method has two adjustable parameters: the proximity parameter θ and number of tree levels k (tree depth).

The first parameter allows changing ratio between the accuracy and computational complexity. All numerical results, presented in the paper, are obtained with relative error not exceeding 0.2 % comparing with the "direct" calculation according to the (1). The number of tree levels effects only the computational complexity, and for every particular problem there is an optimal value, which provides the minimal computational cost.

3 PARALLEL IMPLEMENTATION OF THE BARNES — HUT METHOD

The parallel implementations of the method are developed using both OpenMP and MPI technologies. The terminal cells are split between MPI-processes (or/and OpenMP threads). The stages 1–3 are performed in sequential mode simultaneously by all MPI-processes due to their small contribution in all algorithm.

3.1 Efficiency of the parallel implementation for shared memory system

The first numerical experiment was performed for 18-cores CPU Intel Core i9-7980XE using both OpenMP and MPI technologies. The achieved acceleration is shown in Fig. 2 for the time-consuming problem with large number of vortex elements ($N = 1\,000\,000$).

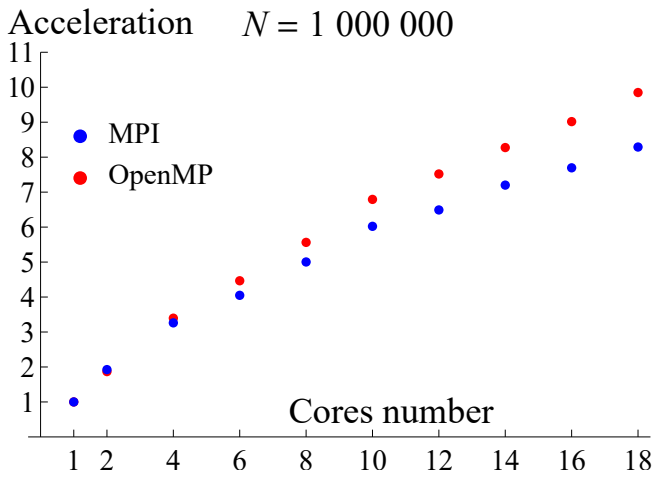


Figure 2: Acceleration of the Barnes — Hut-type method algorithm

It is seen that OpenMP-implementation is more efficient for shared memory systems. Its efficiency for 18 cores is about 55 % while MPI-implementation efficiency is 46 %. Obtained results correspond to the Amdahl's law with 5 % and 7 % of sequential code (for OpenMP and MPI technologies, respectively).

3.2 Efficiency of the parallel implementation for cluster system

For the cluster system there is a possibility of simultaneous usage of both OpenMP and MPI technologies. The numerical results for 3-nodes cluster system with 4-cores processors Intel Core i7-940 are shown in Table 1 for the same problem.

Table 1: Time of computations and acceleration of the Barnes — Hut-type method algorithm for the cluster system

Nodes number	1 OpenMP thread per node		4 OpenMP threads per node		4 MPI process per node	
	Time, sec	Acceleration	Time, sec	Acceleration	Time, sec	Acceleration
1	106.13	1.00	30.84	3.44	33.58	3.16
2	54.02	1.96	16.82	6.31	18.20	5.83
3	36.67	2.89	12.11	**8.76**	13.17	**8.06**

It is seen that the usage of both OpenMP and MPI technologies is more efficient in comparison to the only MPI technology. The efficiency of parallel implementation on 12-cores cluster system is 73 %. Note, that for this case total acceleration comparing to the "direct" sequential algorithm is about 1000 times.

The Barnes — Hut-type method is satisfactory scalable, at the same time its sequential implementation is not very efficient itself. The fact is that this method initially had been developed for the gravitational N-body problem, but we consider two-dimensional problem statement. While in 3D problems the influence of a body on another decreases proportionally to squared distance between them, in 2D case it is inversely proportional to the first degree of the distance only. That's the reason of the other method implementation, which is more efficient for 2D problems.

4 THE FFT-BASED METHOD

This method is based on the possibility of the convolution integral in (1) calculation using Fast Fourier Transform (FFT) technique [6]. As it is shown in [7], the usage of this method "directly" leads to the significant error caused by inaccurate calculated influence from vortex elements located in some neighboring zone. So, the special correction procedure is required. It is based on the linear dependency between the velocity and nodal circulations through some correction matrix $\{V\} = [C]\{\Gamma\}$. In such a way it is possible to exclude the inaccurately calculated influence from the neighboring zone of each cell and add the accurate one, calculated directly using the Biot — Savart law. It was found in numerical experiment, that the optimal neighboring zone size is 3 cell layers [7]. In this case the relative error level is less than 0.2 %; it is acceptable for most applications.

In the flow domain rectangular uniform mesh is introduced, which for simplicity contains $M \times M$ nodes ($M \ll N$). The FFT-based algorithm can be split into 3 blocks:

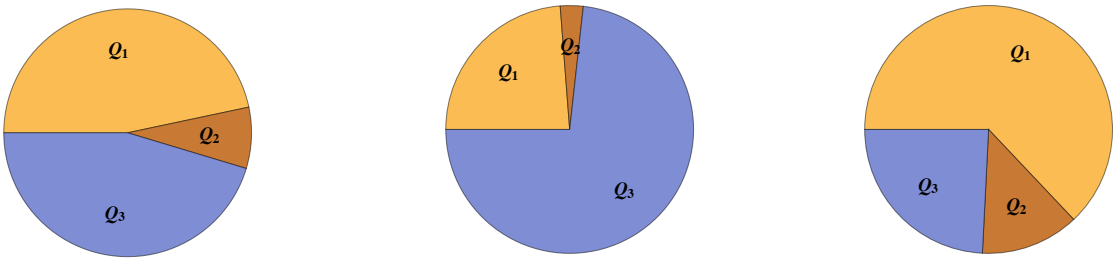
1. Q_1 — nodal circulations calculation (by using the Monaghan's operator M_4 [8]) and correction velocities calculation (which afterwards should be subtracted).

2. Q_2 — convolution integral calculation using the FFT technique.

3. Q_3 — velocities interpolation from the mesh nodes onto the vortex elements and addition the accurately (exactly) calculated vortex influence from the neighboring zone of each cell using the Biot — Savart law.

Note, that the coefficients of the correction matrix $[C]$ depend only on the cell size, so it can be calculated only once at the beginning of the calculation procedure.

The ratio of the operations Q_1, Q_2 and Q_3 significantly depends on the mesh size. Herewith for fixed number of vortex elements N there is some optimal mesh size M when computational complexity of the method is the lowest. The optimal ratio is shown in Fig 3, *a*. But in practice such ratio almost can't be reached due to well-known fact, that for the optimal performance of the fast Fourier transform subroutines the mesh size should be chosen as $M = 2^d$, $d \in \mathbb{N}$. This fact limits the variability of value M, so the real optimal ratio for each problem is deviates from the ratio in Fig. 3, *a*. The examples for two different problems with $N = 100\,000$ and $N = 500\,000$ are shown in Fig. 3, *b*, *c*.



a) Theoretical estimation b) $N = 100\,000$, $M = 128$ c) $N = 500\,000$, $M = 512$

Figure 3: An optimal ratios of the FFT-based algorithm operations

Assuming that the number of vortex elements increases in time, it is important to determine when the mesh size should be doubled. The time of calculations for different time-consuming problems is shown in Fig. 4. The calculations were performed for two mesh sizes: $M = 256$ and $M = 512$.

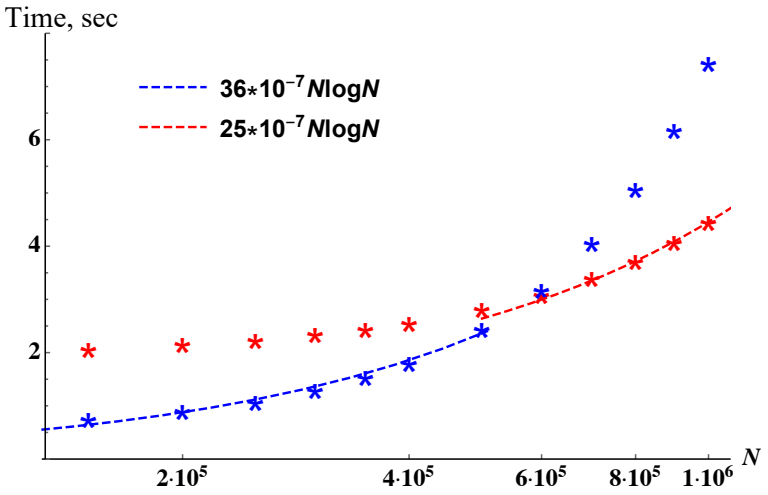


Figure 4: Computational time for different problems (number of vortex elements N); blue asterisks correspond to calculations with $M = 256$; red ones — with $M = 512$

It is seen that the FFT-based method has logarithmic computational complexity $O(N \log N)$, when the mesh size is optimal; the change-over of the mesh size should be performed from $M = 256$ to $M = 512$ when $N > 500\,000$.

If the number of vortex elements is large ($N = 1\,000\,000$), the vortex influence calculation using the "direct" method takes about 3 hours (at one time step); the Barnes — Hut-type method takes 60 seconds and the FFT-based method takes only 4.5 seconds. Thus, we obtain the acceleration more than 2000 times.

5 PARALLEL IMPLEMENTATION OF THE FFT-BASED METHOD

The simulation of the vortex elements movement requires the velocities calculation at every time step. Besides the fact that the number of vortex elements in real problems can exceed hundreds of thousands, the number of time steps can also be tens or even hundreds of thousands, therefore any possible acceleration of calculations is required. For this purpose, assuming that all modern processors are multi-core, the parallel implementation of the above mentioned fast method algorithm is developed.

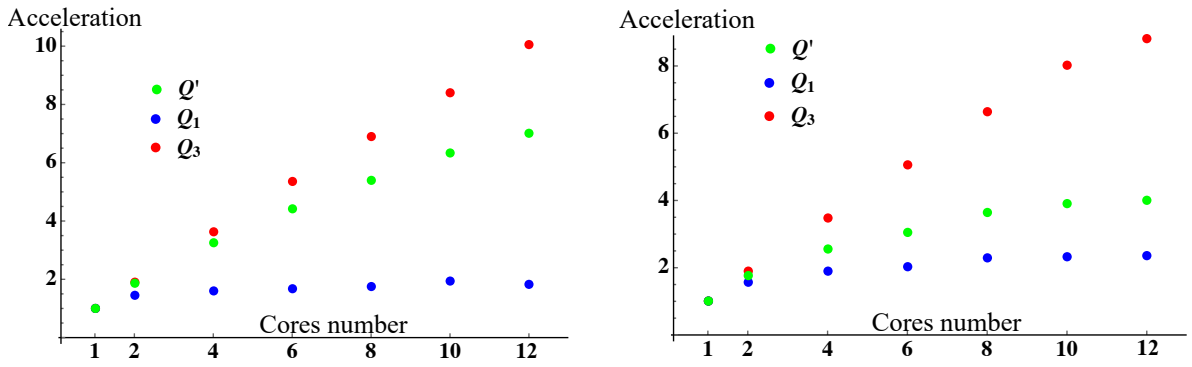
5.1 Parallel implementation using OpenMP technology

The following stages of the algorithm are implemented in parallel mode using OpenMP technology:

1. Mesh cells initialization.
2. Calculation of the matrix $\{\Gamma\}$ of nodal circulations.
3. Velocities interpolation from the mesh nodes onto vortex elements.
4. Calculation of the influence in the neighboring zone directly according to the Biot — Savart law.
5. Summation of the influences calculated approximately and exactly.

As noted earlier, the mesh size M significantly effects the numerical complexity of operations Q_1, Q_2 and Q_3 of the FFT-based method. The operation Q_2 is implemented sequentially, and it takes the most part of the sequential code in the whole algorithm. Its numerical complexity depends only on the mesh size, so the ratio of sequential code can vary. The parallel implementations of the operations Q_1 and Q_3 are developed.

There is an relationship between these two blocks of operations: increasing contribution of one of them leads to decreasing of the other. For small values of M the operation Q_3 preponderates (since in this case the neighboring zone is rather large and it contains large number of vortex elements). Therefore, the acceleration of this operation should increase for coarser mesh and decrease at the mesh refinement. For the operation Q_1 the situation is opposite. So, for optimal mesh size the acceleration will be quite moderate. The numerical results for the problem with $N = 1\,000\,000$ vortex elements prove these estimations (Fig. 5). Here Q' is total numerical complexity of the parallelized operations Q_1 and Q_3. All calculations were performed for the 12-cores CPU Inter Core i9-7980XE.



a) $M = 256$ b) $M = 512$

Figure 5: Different operations accelerations obtained using OpenMP-implementation

It is seen that total acceleration (green points) located somehow between accelerations of operations Q_1 and Q_3, approaching to the most time-consuming one. The highest acceleration and the lowest calculation time are achieved for non-optimal mesh $M = 256$ (Table 2). Here Q_{tot} includes all the operations (Q_1, Q_2 and Q_3).

Table 2: Calculation time and acceleration of the FFT-based method operations using OpenMP

	Acceleration			Time, sec		
	M = 256	$M = 512$	$M = 1024$	**M = 256**	$M = 512$	$M = 1024$
Q_1	1.81	2.35	2.79	0.36	0.56	1.69
Q_3	10.05	8.82	7.69	0.61	0.19	0.10
Q'	7.00	4.00	3.12	0.97	0.75	1.79
Q_{tot}	**6.57**	3.05	2.08	**1.05**	1.12	3.24

In sequential mode the optimal mesh consists of $M = 512$ nodes for such problem, however now we obtain the best result for $M = 256$. It means that the mesh choice depends on available computing resources. Obtained accelerations for the mesh $M = 256$ have a good agreement to the Amdahl's law with 7 % of sequential code.

5.2 Parallel implementation using MPI technology

In parallel implementation of the FFT-based method using MPI technology the computational domain is split vertically into rectangular bands; number of such bands corresponds to the number of MPI-processes.

The MPI-implementation includes the following parts:

a) parallel code (every MPI-process performs this code for its mesh domain):

1. Cells initialization.
2. Calculation of the circulation matrix $\{\Gamma\}$ at mesh nodes.
3. Correction velocities calculation.

4. Velocities interpolation from the mesh nodes onto vortex elements.
5. Calculation of the neighboring zone influence using the Biot — Savart law.
6. Summation of the influences calculated approximately and exactly.

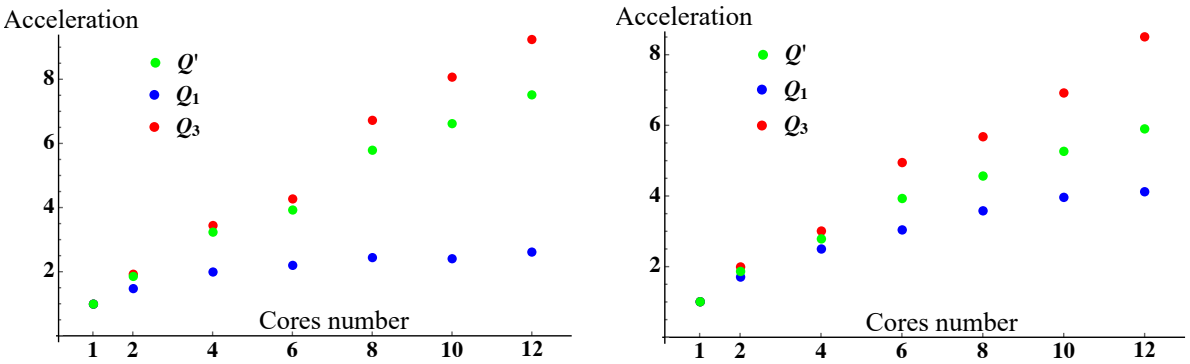
b) data exchange:

1. Transferring the information from the shadow edges. It includes nodal circulations $\{\Gamma\}$ (1 cell layer) and correction velocities (3 cell layers).
2. Gathering all nodal circulations on the master-process for the convolution integral calculation using the FFT (non-blocking data transfer is used).
3. Transferring of the vortex elements data for boundary cells of each process for correct calculation of the vortex influence in neighboring zone of such cells.

c) sequential code:

1. Convolution integral calculation using the Fast Fourier Transform technique.

The numerical results are similar to the previous section (OpenMP implementation). They are shown in Fig. 6 for the same problem with $N = 1\,000\,000$ vortex elements.



a) $M = 256$ b) $M = 512$

Figure 6: Different operations accelerations obtained using MPI-implementation

It is seen again, that the total acceleration for $Q' = Q_1 + Q_3$ (green points) is located somehow between accelerations for Q_1 and Q_3; the highest acceleration is again achieved on the coarser mesh ($M = 256$), but in this case computational time is less on the optimal mesh for considering problem ($M = 512$). The resulting acceleration and computational time are shown in Table 3. Comparing with the similar results shown in Table 2, it can be seen that the MPI technology is more efficient from the computational time point of view, so it is preferable even for the systems with shared memory.

Numerical experiment results for the mesh with the best acceleration ($M = 256$) are in good agreement with Amdahl's law with 6 % of sequential code.

Table 3: Calculation time and acceleration of the FFT-based method operations using MPI

	Acceleration			Time, sec		
	M = 256	$M = 512$	$M = 1024$	$M = 256$	**M = 512**	$M = 1024$
Q_1	2.59	4.12	6.81	0.24	0.31	0.58
Q_3	9.24	8.51	7.78	0.66	0.21	0.11
Q'	7.49	5.89	6.96	0.90	0.51	0.68
Q_{tot}	**6.97**	3.69	2.50	0.98	**0.92**	2.65

5.3 Efficiency of the parallel implementation for cluster system

For the cluster system both OpenMP and MPI technologies can be applied simultaneously. The numerical results for 3-nodes cluster system with 4-cores CPU Intel Core i7-940 are shown in Table 4 for the same problem.

Table 4: Computational time and acceleration of the FFT-based method on cluster system

Nodes number	1 OpenMP thread per node		4 OpenMP threads per node		4 MPI processes per node	
	Time, sec	Acceleration	Time, sec	Acceleration	Time, sec	Acceleration
1	4.46	1.00	2.57	1.73	1.77	2.52
2	2.80	1.59	1.80	2.47	1.49	2.99
3	2.23	2.00	1.56	**2.86**	1.41	**3.16**

As it was discussed in the previous section, the MPI technology is more efficient. So, despite the data transfer, the maximal acceleration (and minimal time) is obtained using only the MPI technology.

6 COMPARISON WITH THE "DIRECT" METHOD

As noted earlier, the "direct" method is highly scalable. Thus, the calculation with usage GPU Tesla V100 takes about 10 seconds (for $N = 1\,000\,000$), while the direct velocities computation takes about 3 hours. The calculation using the Barnes — Hut-type method in sequential mode takes about 60 seconds, while the FFT-based method in sequential mode takes only 4.5 seconds. Thus, the FFT-based method is more efficient even in sequential mode than "direct" method, been running on the most powerful graphic accelerator nowadays. Considering the parallel implementations of both fast methods, we obtain that for the same problem the Barnes — Hut-type method, being run on multicore CPU requires nearly the same time, that the direct method on GPU. At the same time, the FFT-based method on the same CPU is 10 times faster (it takes only 0.9 seconds).

The numerical experiment was performed for 12-cores CPU Intel Core i9-7980XE. The computational time for "direct" and FFT-based method is shown in Fig. (7). It is seen that for 12 cores the FFT-based method becomes more efficient for $N > 150\,000$. If only 4 cores are used (which the most modern processors have) the FFT-based method is comparable with the "direct" GPU-implementation already for $N = 200\,000$.

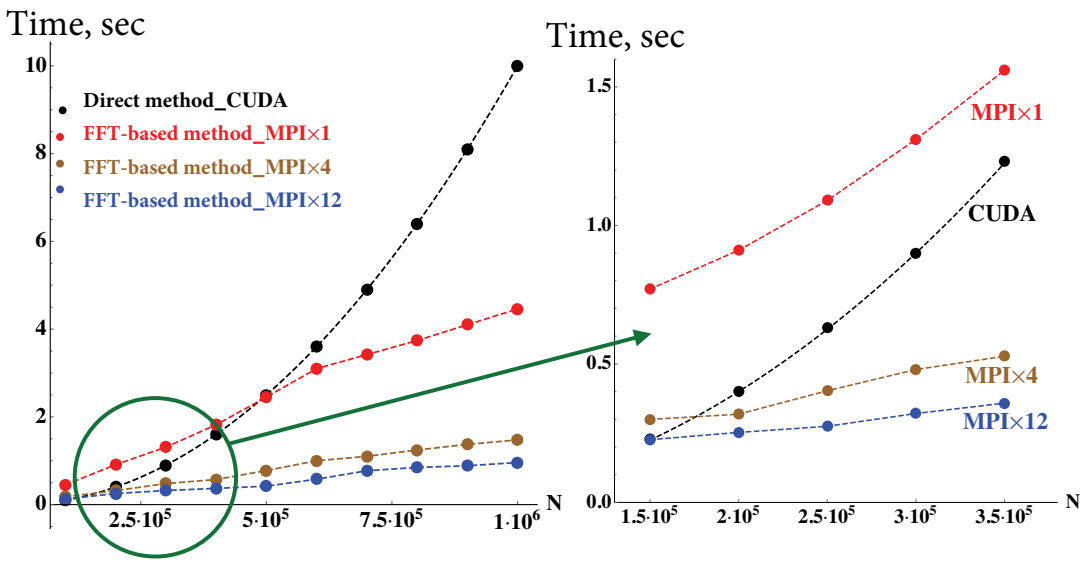


Figure 7: Time of calculations for the FFT-based and "direct" methods

7 CONCLUSIONS

The problem of computational complexity reduction in the algorithms of vortex methods is considered. Two fast approximate methods for vortex influence computation are implemented. Both methods have a logarithmic computational complexity instead of the squared one. Their sequential and parallel implementations are developed. Acceptable relative error for all numerical experiments is less than 0.2 %.

The first method is an analogue of the Barnes-Hut fast method for the gravitational N-body problem. The efficiency of this method for 2D problems is lower in comparison to 3D problems, however, it is scalable and can be parallelized rather easily. For the sequential code the acceleration in comparison to the "direct" (the Biot — Savart law-based) method for the problem with 10^6 vortex elements is about 180 times. Using the OpenMP technology it is possible to achieve additional 10 times acceleration for 18-cores Intel i9-7980XE CPU. For MPI-implementation the acceleration is slightly lower.

The other considered fast method is based on the possibility of convolution integral fast calculation by using the Fast Fourier Transform (FFT) technique with further results correction on the coarse mesh for correct influence accounting of closely-spaced vortex elements. For sequential implementation this method is about 10 times more efficient in comparison to the previous one. For 10^6 vortex elements it is more than 2 000 times faster in comparison to "direct" calculation. At the same time the efficiency of its parallelization is lower, the maximal achieved acceleration for 12-cores CPU is about 6 times (for all the operations, excluding the FFT transform itself).

Even for sequential version the FFT-based method is faster than the "direct" approach, being implemented for GPU architecture for the most powerful graphical accelerator Tesla V100 for number of vortices $N > 500\,000$. Parallel implementation of the FFT method (12 cores) makes it possible to perform one time step for 10^6 vortices within 0.9 seconds, while for Tesla V100 about 10 seconds is required.

Acknowledgement

The research is supported by the Russian Foundation for Basic Research (RFBR), proj. 18-31-20051.

REFERENCES

[1] Cottet, G.H. and Koumoutsakos, P.D. *Vortex methods. Theory and practice.* Cambridge University Press (2000).

[2] Lewis, R.I. Vortex element methods for fluid dynamic analysis of engineering systems. Cambridge University Press (2005).

[3] Anronov, P.R., Guvernuk, S.V. and Dynnikova, G.Ya. *Vortex Methods for Computation of Unsteady Hydrodynamic Loads*. Moscow State University Press (2006).

[4] Kuzmina, K.S., Marchevsky, I.K. and Ryatina, E.P. On CPU and GPU parallelization of VM2D code for 2D flows simulation using vortex method. *Proceedings of the 6th European Conference on Computational Mechanics (ECCM 6) and 7th European Conference on Computational Fluid Dynamics (ECFD 7)* (2018): 2390–2401.

[5] Dynnikova, G.Ya. Fast technique for solving the N-body problem in flow simulation by vortex methods. *Computational Mathematics and Mathematical Physics* (2009) **49**:1389–1396.

[6] Nussbaumer, H.J. *Fast Fourier Transform and Convolution Algorithms*. Springer-Verlag (1982).

[7] Morgenthal, G. and Walther, J.H. An immersed interface method for the Vortex-In-Cell algorithm. *Computers and Structures* (2007) **85**:712–726.

[8] Monaghan, J.J. Extrapolating B-splines for interpolation. *Journal of Computational Physics* (1985) **60**:253-262.